Ex. 13

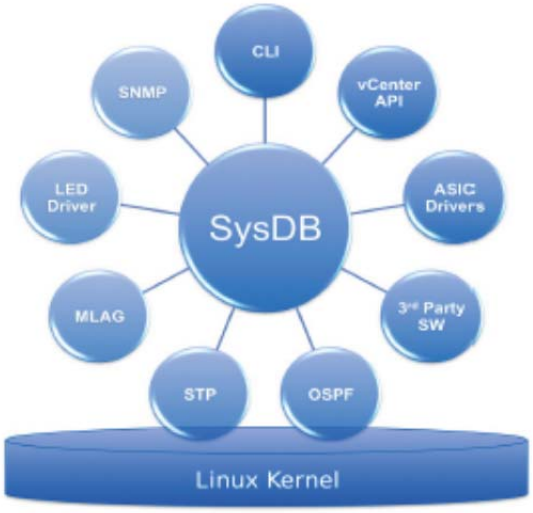
U.S. Patent No. 7,162,537 Infringement Chart

Title: Method and system for externally managing router configuration data in conjunction with a centralized database

Inventor: Pradeep Kathail

Abstract: A method and system for externally managing router configuration data in conjunction with a centralized database subsystem in a router device. The centralized database provides external management registration and unregistration for various managing router subsystems associated with said database system. The centralized database and the subsystems registered for external data management engage in transaction request sequences to provide router data requested by other client subsystems. The arrangement of the various client subsystems associated with the database subsystem allows the client subsystems to remain modular and independent of each other.

U.S. 7,162,537	Arista Products
[1.0] A method for reducing computational overhead in a centralized database system by externally managing router data in conjunction with a centralized database subsystem, said database subsystem operatively coupled for communication	To the extent the preamble is limiting, Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, perform a method for reducing computational overhead in a centralized database system by externally managing router data in conjunction with a centralized database subsystem, said database subsystem operatively coupled for communication with a plurality of router subsystems one of which is a first managing subsystem. The Arista 7150 series, running EOS version 4.13.5F, is an exemplary model for demonstrating Arista's infringement of this patent. *See e.g.,* Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58 ("Arista switches have, at their heart, a database called SysDB. This database contains the state information and settings for the switch, organized in such a way that every module can access it with ease."). *See e.g.,* Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58 ("Not only does EOS separate the networking state from the processing, but drivers, processes, management, and even security patches run in user address space, not in the kernel."). *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 4:

U.S. 7,162,537	Arista Products
with a plurality of router subsystems one of which is a first managing subsystem, comprising:	 **Figure 3**: Arista EOS Architecture *See e.g.*, Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change."). *See, e.g.*, partial output from running "Show process" command on EOS v.4.13-5F:

U.S. 7,162,537	Arista Products

```
localhost#show process
 01:18:00 up 11 min,  1 user,  load average: 0.07, 0.22, 0.24
  PID %CPU %MEM TT       STAT  STARTED     TIME CMD
 3191  2.0  0.0 pts/3    R+   01:17:59 00:00:00 ps -e -o pid,pcpu,pmem,tt,stat,s
 1606  5.6  4.1 ?        S    01:07:18 00:00:36 Sysdb           -d -i --dlopen -
 1932  5.6  2.5 ?        Sl   01:07:46 00:00:34 FocalPoint      -d -i --dlopen -
 1816  1.2  1.4 ?        S    01:07:44 00:00:07 AgentMonitor    -d -i --dlopen -
 1608  1.0  2.8 ?        S    01:07:18 00:00:06 Fru             -d -i --dlopen -
 2058  0.9  1.9 ?        S    01:07:50 00:00:05 PhyAeluros      -d -i --dlopen -
 1860  0.8  2.7 ?        S    01:07:44 00:00:05 /usr/sbin/ribd -N
 1607  0.7  2.2 ?        S    01:07:18 00:00:04 FastClid        -d -i --dlopen -
 1609  0.7  2.1 ?        S    01:07:18 00:00:04 Launcher        -d -i --dlopen -
 2408  0.6  1.9 pts/3    Sl+  01:08:17 00:00:03 Cli [interac    -d -i --dlopen -
 1777  0.5  1.8 ?        S    01:07:44 00:00:03 SuperServer     -d -i --dlopen -
 1605  0.5  1.1 ?        S    01:07:18 00:00:03 ProcMgr-work    -d -i --dlopen -
 1933  0.5  1.7 ?        S    01:07:46 00:00:03 Smbus           -d -i --dlopen -
 1968  0.5  1.7 ?        S    01:07:47 00:00:03 Mdio            -d -i --dlopen -
 1781  0.3  2.0 ?        S    01:07:44 00:00:02 Lag+LacpAgen    -d -i --dlopen -
 1784  0.3  1.7 ?        S    01:07:44 00:00:01 Lldp            -d -i --dlopen -
 2663  0.3  1.6 ?        S    01:08:23 00:00:01 Pmbus           -d -i --dlopen -
 1990  0.2  1.4 ?        S    01:07:48 00:00:01 PhyEthtool      -d -i --dlopen -
 1863  0.2  1.9 ?        S    01:07:44 00:00:01 IgmpSnooping    -d -i --dlopen -
 1851  0.2  1.8 ?        S    01:07:44 00:00:01 Acl             -d -i --dlopen -
 1786  0.2  1.8 ?        S    01:07:44 00:00:01 LacpTxAgent     -d -i --dlopen -
 2041  0.1  1.8 ?        S    01:07:49 00:00:01 XcvrAgent       -d -i --dlopen -
 1915  0.1  1.8 ?        S    01:07:46 00:00:01 Ebra            -d -i --dlopen -
 1848  0.1  1.6 ?        S    01:07:44 00:00:01 Stp             -d -i --dlopen -
 1996  0.1  1.6 ?        S    01:07:48 00:00:00 PowerSupplyD    -d -i --dlopen -
 1802  0.1  1.7 ?        S    01:07:44 00:00:00 Ira             -d -i --dlopen -
    1  0.1  0.3 ?        Ss   01:06:00 00:00:01 /sbin/init
 1792  0.1  1.6 ?        Sl   01:07:44 00:00:00 Aaa             -d -i --dlopen -
 2038  0.1  1.6 ?        S    01:07:49 00:00:00 FanDetector     -d -i --dlopen -
 1846  0.1  1.2 ?        S    01:07:44 00:00:00 Dot1x           -d -i --dlopen -
 1899  0.1  1.7 ?        S    01:07:45 00:00:00 Ucd9012         -d -i --dlopen -
 1864  0.1  1.6 ?        S    01:07:44 00:00:00 Thermostat      -d -i --dlopen -
 1826  0.1  1.7 ?        S    01:07:44 00:00:00 Arp             -d -i --dlopen -
 1805  0.1  1.7 ?        S    01:07:44 00:00:00 LedPolicy       -d -i --dlopen -
 1913  0.1  1.6 ?        S    01:07:46 00:00:00 ScdAgent        -d -i --dlopen -
 1885  0.1  1.5 ?        S    01:07:45 00:00:00 Sb820           -d -i --dlopen -
 1796  0.1  1.6 ?        S    01:07:44 00:00:00 Mirroring       -d -i --dlopen -
 1788  0.1  1.5 ?        S    01:07:44 00:00:00 Bfd             -d -i --dlopen -
 2400  0.1  0.3 ttyS0    Ssl+ 01:08:16 00:00:00 FastCli
 1088  0.1  0.0 ?        S    01:06:38 00:00:00 [kworker/0:2]
 1855  0.1  1.2 ?        S    01:07:44 00:00:00 Fhrp            -d -i --dlopen -
 1994  0.0  1.5 ?        S    01:07:48 00:00:00 Lm95234         -d -i --dlopen -
 1998  0.0  1.5 ?        S    01:07:48 00:00:00 Lm73            -d -i --dlopen -
 1797  0.0  1.2 ?        S    01:07:44 00:00:00 EventMon        -d -i --dlopen -
 1862  0.0  1.6 ?        S    01:07:44 00:00:00 Qos             -d -i --dlopen -
 1795  0.0  1.6 ?        S    01:07:44 00:00:00 PortSec         -d -i --dlopen -
 1887  0.0  1.5 ?        S    01:07:45 00:00:00 PciBus          -d -i --dlopen -
 1971  0.0  1.5 ?        S    01:07:47 00:00:00 Sol             -d -i --dlopen -
 1868  0.0  1.5 ?        S    01:07:44 00:00:00 NetworkTopol    -d -i --dlopen -
 1836  0.0  1.2 ?        S    01:07:44 00:00:00 StpTopology     -d -i --dlopen -
```

U.S. 7,162,537	Arista Products

	See, e.g., source code from \usr\lib\python2.6\site-packages\Sysdbplugin\Stp.py in EOS v.4.13-5F at lns. 22-29:
	`import SysdbPlugin.SysdbLauncher`
	`agentName = 'Stp'`
	`agentCfg = { 'name' : agentName,`
	`'exe' : '/usr/bin/Stp',`
	`'argv' : [],`
	`'heartbeatPeriod' : 30 }`
	`roleName = SysdbPlugin.SysdbLauncher.allSupervisorsRoleName`
	`SysdbPlugin.SysdbLauncher.agentConfigIs( entMan, roleName, agentCfg )`
	
	See e.g., Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 57 ("The operating system for Arista switches is called the Extensible Operating System, or EOS for short . . . Arista switches run Unix natively, but to make them easier for nonprogrammers to understand, EOS makes them look more like traditional (Cisco) networking devices.").
	
	See e.g., Arista At-A-Glance: Extensible Operating System (January 2010) (App. M, Ex. 4) at p. 2 ("Run what you want, and rest assured we have ample CPU capacity available with a dual-core 1.8Ghz AMD x86 CPU and trusted Linux scheduler.").
	
	See e.g., Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change.
	
	All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost.
	
	Sysdb is very much like an event-driven publish/subscribe model. If the state of an agent changes, Sysdb will send

U.S. 7,162,537	Arista Products
	an event notification to that agent, which will then update its local copy. Similarly when the agent writes to the mount point, it only changes its local copy and the write returns immediately. Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent. This centralized database approach to passing state throughout the system, and the automated way the Sysdb code is generated reduces risk and error, improves software feature velocity, and provides flexibility for customers who can use the same APIs to receive notifications from Sysdb to customize and extend switch features.”).
[1.1] a) transmitting a management registration request by said first managing subsystem to said database subsystem, said registration request indicating router configuration data for which said first managing subsystem is requesting to provide external management services, said	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, perform transmitting a management registration request by said first managing subsystem to said database subsystem, said registration request indicating router configuration data for which said first managing subsystem is requesting to provide external management services, said router configuration data managed by said database system and derived from configuration commands supplied by a user and executed by a router configuration subsystem before being stored in said database. *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 (“As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called ‘agents’, by notifying interested processes or agents when there is a change. All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost. Sysdb is very much like an event-driven publish/subscribe model. If the state of an agent changes, Sysdb will send

U.S. 7,162,537	Arista Products
router configuration data managed by said database system and derived from configuration commands supplied by a user and executed by a router configuration subsystem before being stored in said database;	an event notification to that agent, which will then update its local copy. Similarly when the agent writes to the mount point, it only changes its local copy and the write returns immediately. Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent."). *See e.g.,* EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code."). *See, e.g.,* source code from \usr\lib\python2.6\site-packages\Sysdbplugin\Stp.py in EOS v.4.13-5F at lns. 1-35:

U.S. 7,162,537	Arista Products

```
# Copyright (c) 2007-2011 Arista Networks, Inc.  All rights reserved.
# Arista Networks, Inc. Confidential and Proprietary.

import Plugins
import Errdisable

@Plugins.plugin( requires=( 'interface/errdisable/causegroup',
                            'interface/errdisable/cause' ) )
def Plugin( entMan ):
   # Create stp config status and hw.
   entMan.register( 'stp/config', "Stp::Config" )
   entMan.register( 'stp/portMode', "Tac::Dir" )
   entMan.register( 'stp/input/config', "Tac::Dir" )
   entMan.register( 'stp/input/config/cli', "Stp::Input::Config" )
   entMan.register( 'stp/status', "Stp::Status" )
   entMan.register( 'stp/protoStatus', "Stp::ProtoStatus" )
   entMan.register( 'stp/hw', "Stp::Hw" )
   entMan.register( 'stp/counter', "Stp::PortCounterDir" )
   entMan.register( 'stp/standbyStatus', "Stp::StandbyStatus", force=True )
   entMan.register( 'stp/ssoStableControlStatus', "Stp::StableControlStatus" )

   import SysdbPlugin.SysdbLauncher
   agentName = 'Stp'
   agentCfg = { 'name' : agentName,
                'exe' : '/usr/bin/Stp',
                'argv' : [],
                'heartbeatPeriod' : 30 }
   roleName = SysdbPlugin.SysdbLauncher.allSupervisorsRoleName
   SysdbPlugin.SysdbLauncher.agentConfigIs( entMan, roleName, agentCfg )

   entMan.registerLogFacility( 'SPANTREE' )

   # create the bpduguard errdisable CauseStatus entity
   causeDesc = "BPDU received on portfast port."
   Errdisable.ErrdisableCauseGroupInit( entMan, 'bpduguard', False, causeDesc )
```

See, e.g., Arista Configuration Guide v. 4.13.5F (4/2/14) (App. M, Ex. 25) at p. 898 ("**Multiple Spanning Tree**

U.S. 7,162,537	Arista Products

(MST)

Multiple Spanning Tree is enabled by the spanning-tree mode command with the ***mstp*** option. MSTP is the default STP version.

Example

- This command enables Multiple Spanning Tree.

switch (config) #**spanning-tree mode mstp**
switch (config) #") (emphasis in original).

See, e.g., Arista Configuration Guide v. 4.13.5F (4/2/14) (App. M, Ex. 25) at p. 902 ("The switch defines bridge IDs for three MST instances:

- MST 0: 32768 (Priority (32768)+Instance number(0)) and 001c.7301.23de (MAC address)
- MST101: 32869 (Priority (32768)+Instance number(101)) and 001c.7301.23de (MAC address)
- MST102: 32870 (Priority (32768)+Instance number(102)) and 001c.7301.23de (MAC address)

This command displays a table of root bridge information.

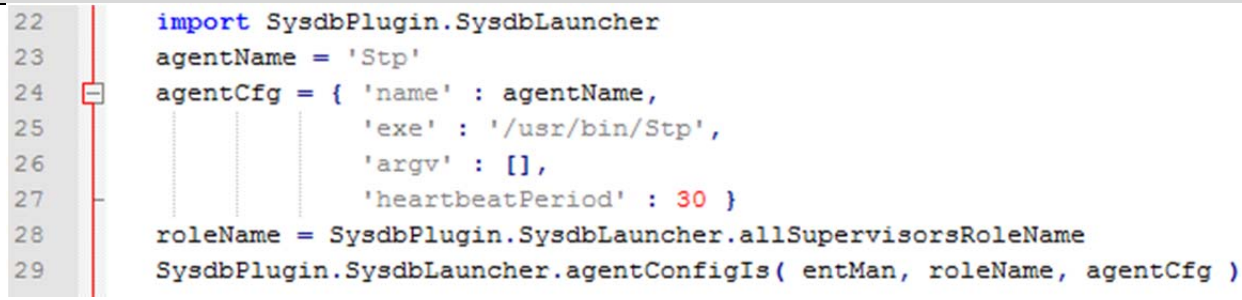
switch>**show spanning-tree root**

Instance	Root ID Priority MAC addr	Root Cost	Hello Time	Max Age	Fwd Dly	Root Port
MST0	32768 001c.7301.23de	0	2	20	15	Po937
MST101	32869 001c.7301.23de	3998	0	0	0	Po909
MST102	32870 001c.7301.23de	3998	0	0	0	Po911

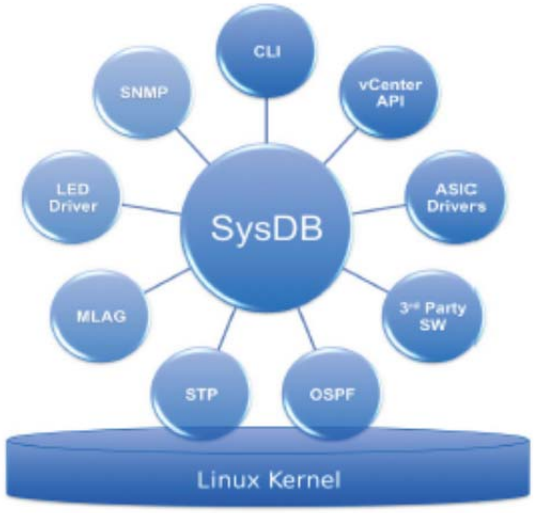
(emphasis in original).

See e.g., Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58-59 ("You can see this in action, though indirectly. When you first log in to an Arista switch, the first thing you might do is issue the show run command. Since even your CLI session is a process with its own user space, the first time you issue the show run command, the process must mount the SysDB database. That takes a second or two, and you may notice the lag. After you get

U.S. 7,162,537	Arista Products
	the output delivered, if you execute the show run command again, it delivers the output much faster because SysDB is already mounted. If you disconnect and then connect again, you'll spawn a new CLI process, which must then mount SysDB once more."). *See e.g.*, Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58-59 ("In fact, if you're impatient enough when first logging in and if you bang on the Enter key, you might be treated to the following message: Arista-7124SX login: **admin** waiting for mounts to complete . . . ok Arista-7124SX>") (emphasis in original).
[1.2] b) receiving said management registration request by said database subsystem; and	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, perform receiving said management registration request by said database subsystem. *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost."). *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent."). *See, e.g.,* source code from \usr\lib\python2.6\site-packages\Sysdbplugin\Stp.py in EOS v.4.13-5F at lns. 22-29:

U.S. 7,162,537	Arista Products
	
[1.3] c) registering said first managing subsystem for external management by said database subsystem.	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, perform registering said first managing subsystem for external management by said managing subsystem. *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost."). *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent."). *See e.g.,* EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code.").

U.S. 7,162,537	Arista Products
[10.0] A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for reducing computational overhead in a centralized database system by externally managing router data in conjunction with a centralized database subsystem, said database subsystem operatively coupled for communication with a plurality	To the extent the preamble is limiting, Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, include a program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for reducing computational overhead in a centralized database system by externally managing router data in conjunction with a centralized database subsystem, said database subsystem operatively coupled for communication with a plurality of router subsystems. The Arista 7150 series, running EOS version 4.13.5F, is an exemplary model for demonstrating Arista's infringement of this patent. *See e.g.,* Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58 ("Arista switches have, at their heart, a database called SysDB. This database contains the state information and settings for the switch, organized in such a way that every module can access it with ease."). *See e.g.,* Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58 ("Not only does EOS separate the networking state from the processing, but drivers, processes, management, and even security patches run in user address space, not in the kernel."). *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 4:

U.S. 7,162,537	Arista Products
of router subsystems one of which is a first managing subsystem, said method comprising:	 **Figure 3**: Arista EOS Architecture *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change."). *See, e.g.*, output from running "Show process" command on EOS v.4.13-5F:

U.S. 7,162,537	Arista Products

```
localhost#show process
 01:18:00 up 11 min,  1 user,  load average: 0.07, 0.22, 0.24
  PID %CPU %MEM TT       STAT  STARTED     TIME CMD
 3191  2.0  0.0 pts/3    R+   01:17:59 00:00:00 ps -e -o pid,pcpu,pmem,tt,stat,s
 1606  5.6  4.1 ?        S    01:07:18 00:00:36 Sysdb           -d -i --dlopen -
 1932  5.6  2.5 ?        Sl   01:07:46 00:00:34 FocalPoint      -d -i --dlopen -
 1816  1.2  1.4 ?        S    01:07:44 00:00:07 AgentMonitor    -d -i --dlopen -
 1608  1.0  2.8 ?        S    01:07:18 00:00:06 Fru             -d -i --dlopen -
 2058  0.9  1.9 ?        S    01:07:50 00:00:05 PhyAeluros      -d -i --dlopen -
 1860  0.8  2.7 ?        S    01:07:44 00:00:05 /usr/sbin/ribd -N
 1607  0.7  2.2 ?        S    01:07:18 00:00:04 FastClid        -d -i --dlopen -
 1609  0.7  2.1 ?        S    01:07:18 00:00:04 Launcher        -d -i --dlopen -
 2408  0.6  1.9 pts/3    Sl+  01:08:17 00:00:03 Cli [interac    -d -i --dlopen -
 1777  0.5  1.8 ?        S    01:07:44 00:00:03 SuperServer     -d -i --dlopen -
 1605  0.5  1.1 ?        S    01:07:18 00:00:03 ProcMgr-work    -d -i --dlopen -
 1933  0.5  1.7 ?        S    01:07:46 00:00:03 Smbus           -d -i --dlopen -
 1968  0.5  1.7 ?        S    01:07:47 00:00:03 Mdio            -d -i --dlopen -
 1781  0.3  2.0 ?        S    01:07:44 00:00:02 Lag+LacpAgen    -d -i --dlopen -
 1784  0.3  1.7 ?        S    01:07:44 00:00:01 Lldp            -d -i --dlopen -
 2663  0.3  1.6 ?        S    01:08:23 00:00:01 Pmbus           -d -i --dlopen -
 1990  0.2  1.4 ?        S    01:07:48 00:00:01 PhyEthtool      -d -i --dlopen -
 1863  0.2  1.9 ?        S    01:07:44 00:00:01 IgmpSnooping    -d -i --dlopen -
 1851  0.2  1.8 ?        S    01:07:44 00:00:01 Acl             -d -i --dlopen -
 1786  0.2  1.8 ?        S    01:07:44 00:00:01 LacpTxAgent     -d -i --dlopen -
 2041  0.1  1.8 ?        S    01:07:49 00:00:01 XcvrAgent       -d -i --dlopen -
 1915  0.1  1.8 ?        S    01:07:46 00:00:01 Ebra            -d -i --dlopen -
 1848  0.1  1.6 ?        S    01:07:44 00:00:01 Stp             -d -i --dlopen -
 1996  0.1  1.6 ?        S    01:07:48 00:00:00 PowerSupplyD    -d -i --dlopen -
 1802  0.1  1.7 ?        S    01:07:44 00:00:00 Ira             -d -i --dlopen -
    1  0.1  0.3 ?        Ss   01:06:00 00:00:01 /sbin/init
 1792  0.1  1.6 ?        Sl   01:07:44 00:00:00 Aaa             -d -i --dlopen -
 2038  0.1  1.6 ?        S    01:07:49 00:00:00 FanDetector     -d -i --dlopen -
 1846  0.1  1.2 ?        S    01:07:44 00:00:00 Dot1x           -d -i --dlopen -
 1899  0.1  1.7 ?        S    01:07:45 00:00:00 Ucd9012         -d -i --dlopen -
 1864  0.1  1.6 ?        S    01:07:44 00:00:00 Thermostat      -d -i --dlopen -
 1826  0.1  1.7 ?        S    01:07:44 00:00:00 Arp             -d -i --dlopen -
 1805  0.1  1.7 ?        S    01:07:44 00:00:00 LedPolicy       -d -i --dlopen -
 1913  0.1  1.6 ?        S    01:07:46 00:00:00 ScdAgent        -d -i --dlopen -
 1885  0.1  1.5 ?        S    01:07:45 00:00:00 Sb820           -d -i --dlopen -
 1796  0.1  1.6 ?        S    01:07:44 00:00:00 Mirroring       -d -i --dlopen -
 1788  0.1  1.5 ?        S    01:07:44 00:00:00 Bfd             -d -i --dlopen -
 2400  0.1  0.3 ttyS0    Ssl+ 01:08:16 00:00:00 FastCli
 1088  0.1  0.0 ?        S    01:06:38 00:00:00 [kworker/0:2]
 1855  0.1  1.2 ?        S    01:07:44 00:00:00 Fhrp            -d -i --dlopen -
 1994  0.0  1.5 ?        S    01:07:48 00:00:00 Lm95234         -d -i --dlopen -
 1998  0.0  1.5 ?        S    01:07:48 00:00:00 Lm73            -d -i --dlopen -
 1797  0.0  1.2 ?        S    01:07:44 00:00:00 EventMon        -d -i --dlopen -
 1862  0.0  1.6 ?        S    01:07:44 00:00:00 Qos             -d -i --dlopen -
 1795  0.0  1.6 ?        S    01:07:44 00:00:00 PortSec         -d -i --dlopen -
 1887  0.0  1.5 ?        S    01:07:45 00:00:00 PciBus          -d -i --dlopen -
 1971  0.0  1.5 ?        S    01:07:47 00:00:00 Sol             -d -i --dlopen -
 1868  0.0  1.5 ?        S    01:07:44 00:00:00 NetworkTopol    -d -i --dlopen -
 1836  0.0  1.2 ?        S    01:07:44 00:00:00 StpTopology     -d -i --dlopen -
```

U.S. 7,162,537	Arista Products

	See, e.g., source code from \usr\lib\python2.6\site-packages\Sysdbplugin\Stp.py in EOS v.4.13-5F at lns. 22-29:
	`import SysdbPlugin.SysdbLauncher`
	`agentName = 'Stp'`
	`agentCfg = { 'name' : agentName,`
	`'exe' : '/usr/bin/Stp',`
	`'argv' : [],`
	`'heartbeatPeriod' : 30 }`
	`roleName = SysdbPlugin.SysdbLauncher.allSupervisorsRoleName`
	`SysdbPlugin.SysdbLauncher.agentConfigIs( entMan, roleName, agentCfg )`
	
	See e.g., Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 57 ("The operating system for Arista switches is called the Extensible Operating System, or EOS for short . . . Arista switches run Unix natively, but to make them easier for nonprogrammers to understand, EOS makes them look more like traditional (Cisco) networking devices.").
	
	See e.g., Arista At-A-Glance: Extensible Operating System (January 2010) (App. M, Ex. 4) at p. 2 ("Run what you want, and rest assured we have ample CPU capacity available with a dual-core 1.8Ghz AMD x86 CPU and trusted Linux scheduler.").
	
	See e.g., Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change.
	
	All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost.
	
	Sysdb is very much like an event-driven publish/subscribe model. If the state of an agent changes, Sysdb will send

U.S. 7,162,537	Arista Products
	an event notification to that agent, which will then update its local copy. Similarly when the agent writes to the mount point, it only changes its local copy and the write returns immediately. Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent. This centralized database approach to passing state throughout the system, and the automated way the Sysdb code is generated reduces risk and error, improves software feature velocity, and provides flexibility for customers who can use the same APIs to receive notifications from Sysdb to customize and extend switch features.”).
[10.1] (a) transmitting a management registration request by said first managing subsystem to said database subsystem, said registration request indicating router configuration data for which said first managing subsystem is requesting to provide external management services, said	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, perform transmitting a management registration request by said first managing subsystem to said database subsystem, said registration request indicating router configuration data for which said first managing subsystem is requesting to provide external management services, said router configuration data managed by said database system and derived from configuration commands supplied by a user and executed by a router configuration subsystem before being stored in said database. *See e.g.*, Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 (“As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called ‘agents’, by notifying interested processes or agents when there is a change. All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost. Sysdb is very much like an event-driven publish/subscribe model. If the state of an agent changes, Sysdb will send

U.S. 7,162,537	Arista Products
router configuration data managed by said database system and derived from configuration commands supplied by a user and executed by a router configuration subsystem before being stored in said database;	an event notification to that agent, which will then update its local copy. Similarly when the agent writes to the mount point, it only changes its local copy and the write returns immediately. Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent."). *See e.g.,* EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code."). *See, e.g.,* source code from \usr\lib\python2.6\site-packages\Sysdbplugin\Stp.py in EOS v.4.13-5F at lns. 1-35:

U.S. 7,162,537	Arista Products

```
# Copyright (c) 2007-2011 Arista Networks, Inc.  All rights reserved.
# Arista Networks, Inc. Confidential and Proprietary.

import Plugins
import Errdisable

@Plugins.plugin( requires=( 'interface/errdisable/causegroup',
                            'interface/errdisable/cause' ) )
def Plugin( entMan ):
   # Create stp config status and hw.
   entMan.register( 'stp/config', "Stp::Config" )
   entMan.register( 'stp/portMode', "Tac::Dir" )
   entMan.register( 'stp/input/config', "Tac::Dir" )
   entMan.register( 'stp/input/config/cli', "Stp::Input::Config" )
   entMan.register( 'stp/status', "Stp::Status" )
   entMan.register( 'stp/protoStatus', "Stp::ProtoStatus" )
   entMan.register( 'stp/hw', "Stp::Hw" )
   entMan.register( 'stp/counter', "Stp::PortCounterDir" )
   entMan.register( 'stp/standbyStatus', "Stp::StandbyStatus", force=True )
   entMan.register( 'stp/ssoStableControlStatus', "Stp::StableControlStatus" )

   import SysdbPlugin.SysdbLauncher
   agentName = 'Stp'
   agentCfg = { 'name' : agentName,
                'exe' : '/usr/bin/Stp',
                'argv' : [],
                'heartbeatPeriod' : 30 }
   roleName = SysdbPlugin.SysdbLauncher.allSupervisorsRoleName
   SysdbPlugin.SysdbLauncher.agentConfigIs( entMan, roleName, agentCfg )

   entMan.registerLogFacility( 'SPANTREE' )

   # create the bpduguard errdisable CauseStatus entity
   causeDesc = "BPDU received on portfast port."
   Errdisable.ErrdisableCauseGroupInit( entMan, 'bpduguard', False, causeDesc )
```

See, e.g., Arista Configuration Guide v. 4.13.5F (4/2/14) (App. M, Ex. 25) at p. 898 ("**Multiple Spanning Tree**

<table>
<tr><th>U.S. 7,162,537</th><th>Arista Products</th></tr>
<tr><td></td><td>
(MST)

Multiple Spanning Tree is enabled by the spanning-tree mode command with the mstp option. MSTP is the default STP version.

Example
• This command enables Multiple Spanning Tree.

switch (config) #spanning-tree mode mstp
switch (config) #") (emphasis in original).

See, e.g., Arista Configuration Guide v. 4.13.5F (4/2/14) (App. M, Ex. 25) at p. 902 ("The switch defines bridge IDs for three MST instances:
• MST 0: 32768 (Priority (32768)+Instance number(0)) and 001c.7301.23de (MAC address)
• MST101: 32869 (Priority (32768)+Instance number(101)) and 001c.7301.23de (MAC address)
• MST102: 32870 (Priority (32768)+Instance number(102)) and 001c.7301.23de (MAC address)

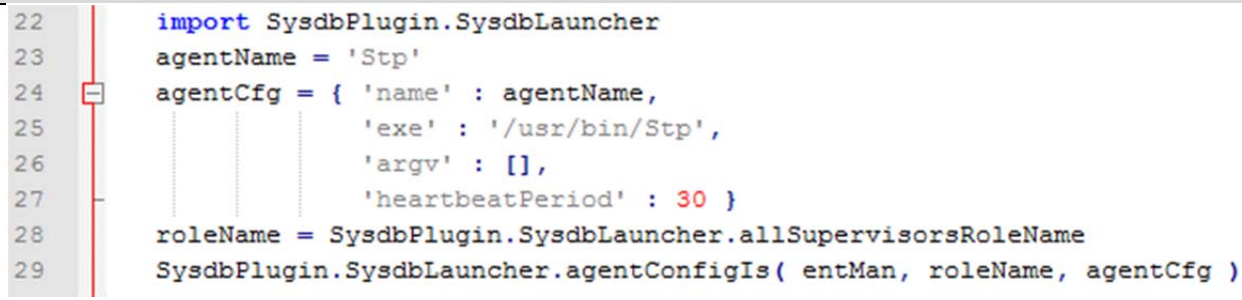
This command displays a table of root bridge information.

switch>show spanning-tree root

<table>
<tr><th>Instance</th><th>Root ID
Priority MAC addr</th><th>Root
Cost</th><th>Hello
Time</th><th>Max
Age</th><th>Fwd
Dly</th><th>Root Port</th></tr>
<tr><td>MST0</td><td>32768 001c.7301.23de</td><td>0</td><td>2</td><td>20</td><td>15</td><td>Po937</td></tr>
<tr><td>MST101</td><td>32869 001c.7301.23de</td><td>3998</td><td>0</td><td>0</td><td>0</td><td>Po909</td></tr>
<tr><td>MST102</td><td>32870 001c.7301.23de</td><td>3998</td><td>0</td><td>0</td><td>0</td><td>Po911</td></tr>
</table>
(emphasis in original).

See e.g., Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58-59 ("You can see this in action, though indirectly. When you first log in to an Arista switch, the first thing you might do is issue the show run command. Since even your CLI session is a process with its own user space, the first time you issue the show run command, the process must mount the SysDB database. That takes a second or two, and you may notice the lag. After you get
</td></tr>
</table>

U.S. 7,162,537	Arista Products
	the output delivered, if you execute the show run command again, it delivers the output much faster because SysDB is already mounted. If you disconnect and then connect again, you'll spawn a new CLI process, which must then mount SysDB once more."). *See e.g.*, Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58-59 ("In fact, if you're impatient enough when first logging in and if you bang on the Enter key, you might be treated to the following message: Arista-7124SX login: **admin** waiting for mounts to complete . . . ok Arista-7124SX>") (emphasis in original).
[10.2] (b) receiving said management registration request by said database subsystem; and	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, perform receiving said management registration request by said database subsystem. *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost."). *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent."). *See, e.g.,* source code from \usr\lib\python2.6\site-packages\Sysdbplugin\Stp.py in EOS v.4.13-5F at lns. 22-29:

U.S. 7,162,537	Arista Products
	
[10.3] (c) registering said first managing subsystem for external management by said managing subsystem.	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, perform registering said first managing subsystem for external management by said managing subsystem. *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost."). *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent."). *See e.g.,* EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code.").
[redacted]	[redacted]

U.S. 7,162,537	Arista Products
[19.0] In a router device having a processor and memory, a router operating system executing within said memory comprising:	To the extent the preamble is limiting, Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, include, in a router device having a processor and memory, a router operating system executing within said memory. The Arista 7150 series, running EOS version 4.13.5F, is an exemplary model for demonstrating Arista's infringement of this patent. *See e.g.* 7150 Series 1/10 GbE SFP Ultra Low Latency Switch: Data Sheet (2012) (App. M, Ex. 1) at p. 6: CPU Dual-Core x86 System Memory 4 Gigabytes Flash Storage Memory 2 Gigabytes
[19.1] (a) a database subsystem;	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, include a database subsystem. *See e.g.,* Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58 ("Arista switches have, at their heart, a database called SysDB. This database contains the state information and settings for the switch, organized in such a way that every module can access it with ease."). *See e.g.,* Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58 ("Not only does EOS separate the networking state from the processing, but drivers, processes, management, and even security patches run in user address space, not in the kernel."). *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 4:

U.S. 7,162,537	Arista Products

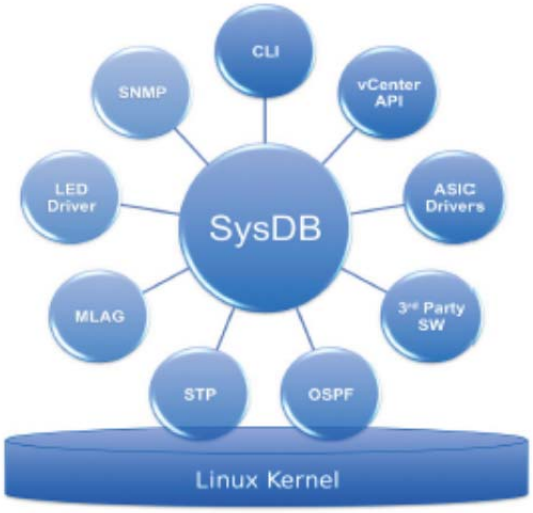


Figure 3: Arista EOS Architecture

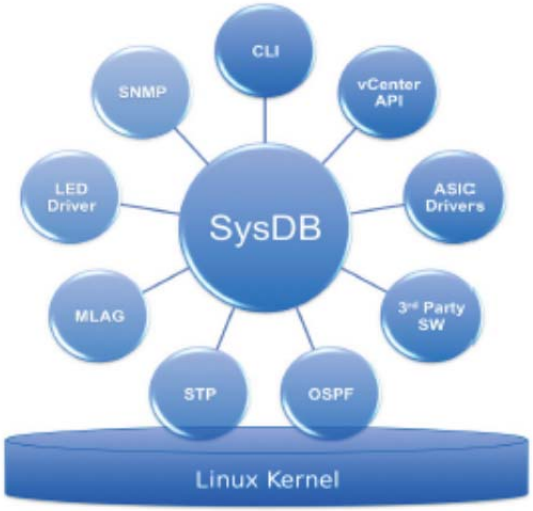
See e.g., Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change.").

See e.g., EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code.").

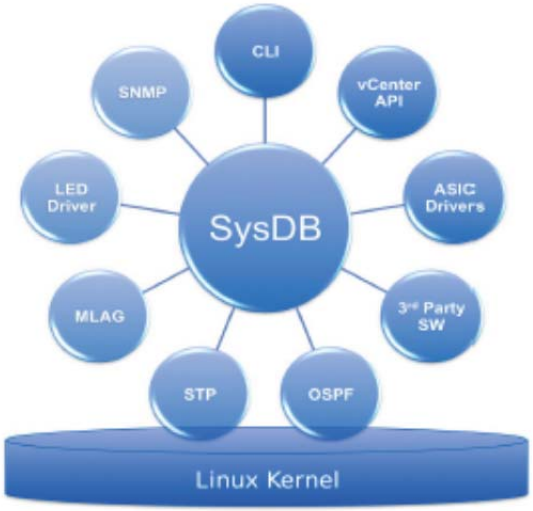
U.S. 7,162,537	Arista Products
	See, e.g., output from running "Show process" command on EOS v.4.13-5F:

U.S. 7,162,537	Arista Products

```
localhost#show process
 01:18:00 up 11 min,  1 user,  load average: 0.07, 0.22, 0.24
  PID %CPU %MEM TT       STAT  STARTED     TIME CMD
 3191  2.0  0.0 pts/3    R+   01:17:59 00:00:00 ps -e -o pid,pcpu,pmem,tt,stat,s
 1606  5.6  4.1 ?        S    01:07:18 00:00:36 Sysdb           -d -i --dlopen -
 1932  5.6  2.5 ?        Sl   01:07:46 00:00:34 FocalPoint      -d -i --dlopen -
 1816  1.2  1.4 ?        S    01:07:44 00:00:07 AgentMonitor    -d -i --dlopen -
 1608  1.0  2.8 ?        S    01:07:18 00:00:06 Fru             -d -i --dlopen -
 2058  0.9  1.9 ?        S    01:07:50 00:00:05 PhyAeluros      -d -i --dlopen -
 1860  0.8  2.7 ?        S    01:07:44 00:00:05 /usr/sbin/ribd -N
 1607  0.7  2.2 ?        S    01:07:18 00:00:04 FastClid        -d -i --dlopen -
 1609  0.7  2.1 ?        S    01:07:18 00:00:04 Launcher        -d -i --dlopen -
 2408  0.6  1.9 pts/3    Sl+  01:08:17 00:00:03 Cli [interac    -d -i --dlopen -
 1777  0.5  1.8 ?        S    01:07:44 00:00:03 SuperServer     -d -i --dlopen -
 1605  0.5  1.1 ?        S    01:07:18 00:00:03 ProcMgr-work    -d -i --dlopen -
 1933  0.5  1.7 ?        S    01:07:46 00:00:03 Smbus           -d -i --dlopen -
 1968  0.5  1.7 ?        S    01:07:47 00:00:03 Mdio            -d -i --dlopen -
 1781  0.3  2.0 ?        S    01:07:44 00:00:02 Lag+LacpAgen    -d -i --dlopen -
 1784  0.3  1.7 ?        S    01:07:44 00:00:01 Lldp            -d -i --dlopen -
 2663  0.3  1.6 ?        S    01:08:23 00:00:01 Pmbus           -d -i --dlopen -
 1990  0.2  1.4 ?        S    01:07:48 00:00:01 PhyEthtool      -d -i --dlopen -
 1863  0.2  1.9 ?        S    01:07:44 00:00:01 IgmpSnooping    -d -i --dlopen -
 1851  0.2  1.8 ?        S    01:07:44 00:00:01 Acl             -d -i --dlopen -
 1786  0.2  1.8 ?        S    01:07:44 00:00:01 LacpTxAgent     -d -i --dlopen -
 2041  0.1  1.8 ?        S    01:07:49 00:00:01 XcvrAgent       -d -i --dlopen -
 1915  0.1  1.8 ?        S    01:07:46 00:00:01 Ebra            -d -i --dlopen -
 1848  0.1  1.6 ?        S    01:07:44 00:00:01 Stp             -d -i --dlopen -
 1996  0.1  1.6 ?        S    01:07:48 00:00:00 PowerSupplyD    -d -i --dlopen -
 1802  0.1  1.7 ?        S    01:07:44 00:00:00 Ira             -d -i --dlopen -
    1  0.1  0.3 ?        Ss   01:06:00 00:00:01 /sbin/init
 1792  0.1  1.6 ?        Sl   01:07:44 00:00:00 Aaa             -d -i --dlopen -
 2038  0.1  1.6 ?        S    01:07:49 00:00:00 FanDetector     -d -i --dlopen -
 1846  0.1  1.2 ?        S    01:07:44 00:00:00 Dot1x           -d -i --dlopen -
 1899  0.1  1.7 ?        S    01:07:45 00:00:00 Ucd9012         -d -i --dlopen -
 1864  0.1  1.6 ?        S    01:07:44 00:00:00 Thermostat      -d -i --dlopen -
 1826  0.1  1.7 ?        S    01:07:44 00:00:00 Arp             -d -i --dlopen -
 1805  0.1  1.7 ?        S    01:07:44 00:00:00 LedPolicy       -d -i --dlopen -
 1913  0.1  1.6 ?        S    01:07:46 00:00:00 ScdAgent        -d -i --dlopen -
 1885  0.1  1.5 ?        S    01:07:45 00:00:00 Sb820           -d -i --dlopen -
 1796  0.1  1.6 ?        S    01:07:44 00:00:00 Mirroring       -d -i --dlopen -
 1788  0.1  1.5 ?        S    01:07:44 00:00:00 Bfd             -d -i --dlopen -
 2400  0.1  0.3 ttyS0    Ssl+ 01:08:16 00:00:00 FastCli
 1088  0.1  0.0 ?        S    01:06:38 00:00:00 [kworker/0:2]
 1855  0.1  1.2 ?        S    01:07:44 00:00:00 Fhrp            -d -i --dlopen -
 1994  0.0  1.5 ?        S    01:07:48 00:00:00 Lm95234         -d -i --dlopen -
 1998  0.0  1.5 ?        S    01:07:48 00:00:00 Lm73            -d -i --dlopen -
 1797  0.0  1.2 ?        S    01:07:44 00:00:00 EventMon        -d -i --dlopen -
 1862  0.0  1.6 ?        S    01:07:44 00:00:00 Qos             -d -i --dlopen -
 1795  0.0  1.6 ?        S    01:07:44 00:00:00 PortSec         -d -i --dlopen -
 1887  0.0  1.5 ?        S    01:07:45 00:00:00 PciBus          -d -i --dlopen -
 1971  0.0  1.5 ?        S    01:07:47 00:00:00 Sol             -d -i --dlopen -
 1868  0.0  1.5 ?        S    01:07:44 00:00:00 NetworkTopol    -d -i --dlopen -
 1836  0.0  1.2 ?        S    01:07:44 00:00:00 StpTopology     -d -i --dlopen -
```

U.S. 7,162,537	Arista Products
[19.2] (b) a plurality of client subsystems, each operatively coupled for communication to said database subsystem, one of said client subsystems configured as a managing subsystem to externally manage router data upon issuing a management request to said database subsystem; and	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, include a plurality of client subsystems, each operatively coupled for communication to said database subsystem, one of said client subsystems configured as a managing subsystem to externally manage router data upon issuing a management request to said database subsystem. *See e.g.*, Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 4:  **Figure 3**: Arista EOS Architecture *See e.g.*, Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state.

U.S. 7,162,537	Arista Products
	However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change."). *See e.g.,* Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change. All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost. Sysdb is very much like an event-driven publish/subscribe model. If the state of an agent changes, Sysdb will send an event notification to that agent, which will then update its local copy. Similarly when the agent writes to the mount point, it only changes its local copy and the write returns immediately. Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent. This centralized database approach to passing state throughout the system, and the automated way the Sysdb code is generated reduces risk and error, improves software feature velocity, and provides flexibility for customers who can use the same APIs to receive notifications from Sysdb to customize and extend switch features.").
[19.3] (c) a database operatively coupled to said database	Arista switches, including at least the 7010, 7048, 7050, 7050X, 7150, 7250X, 7280E, 7300, 7300X, and 7500E series models, and/or Arista EOS, including at least version 4.13.5F, include a database operatively coupled to said database subsystem, said database configured to store router configuration data and delegate management of router configuration data to a management subsystem that requests to manage router configuration data, said router configuration data managed by said database system and derived from configuration commands supplied by a user

U.S. 7,162,537	Arista Products
subsystem, said database configured to store router configuration data and delegate management of router configuration data to a management subsystem that requests to manage router configuration data, said router configuration data managed by said database system and derived from configuration commands supplied by a user and executed by a router configuration subsystem before being stored in said	and executed by a router configuration subsystem before being stored in said database. *See e.g.*, Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58 ("Arista switches have, at their heart, a database called SysDB. This database contains the state information and settings for the switch, organized in such a way that every module can access it with ease."). *See e.g.*, Gary A. Donahue, Arista Warrior (2013) (App. M, Ex. 17) at p. 58 ("Not only does EOS separate the networking state from the processing, but drivers, processes, management, and even security patches run in user address space, not in the kernel."). *See e.g.*, Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex. 14) at p. 4:  Figure 3: Arista EOS Architecture *See e.g.*, Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex.

U.S. 7,162,537	Arista Products
database.	14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change. All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost. Sysdb is very much like an event-driven publish/subscribe model. If the state of an agent changes, Sysdb will send an event notification to that agent, which will then update its local copy. Similarly when the agent writes to the mount point, it only changes its local copy and the write returns immediately. Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent. This centralized database approach to passing state throughout the system, and the automated way the Sysdb code is generated reduces risk and error, improves software feature velocity, and provides flexibility for customers who can use the same APIs to receive notifications from Sysdb to customize and extend switch features."). *See e.g.,* EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code."). *See e.g.,* EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent

U.S. 7,162,537	Arista Products

to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code.").

See, e.g., Arista Configuration Guide v. 4.13.5F (4/2/14) (App. M, Ex. 25) at p. 898 ("**Multiple Spanning Tree (MST)**

Multiple Spanning Tree is enabled by the spanning-tree mode command with the ***mstp*** option. MSTP is the default STP version.

Example

- This command enables Multiple Spanning Tree.

switch (config) #**spanning-tree mode mstp**
switch (config) #") (emphasis in original).

See, e.g., Arista Configuration Guide v. 4.13.5F (4/2/14) (App. M, Ex. 25) at p. 902 ("The switch defines bridge IDs for three MST instances:

- MST 0: 32768 (Priority (32768)+Instance number(0)) and 001c.7301.23de (MAC address)
- MST101: 32869 (Priority (32768)+Instance number(101)) and 001c.7301.23de (MAC address)
- MST102: 32870 (Priority (32768)+Instance number(102)) and 001c.7301.23de (MAC address)

This command displays a table of root bridge information.

switch>**show spanning-tree root**

Instance	Root ID Priority MAC addr	Root Cost	Hello Time	Max Age	Fwd Dly	Root Port
MST0	32768 001c.7301.23de	0	2	20	15	Po937
MST101	32869 001c.7301.23de	3998	0	0	0	Po909
MST102	32870 001c.7301.23de	3998	0	0	0	Po911

U.S. 7,162,537	Arista Products
	(emphasis in original). *See, e.g.*, source code from \usr\lib\python2.6\site-packages\Sysdbplugin\Stp.py in EOS v.4.13-5F at lns. 1-35:

U.S. 7,162,537	Arista Products

```
# Copyright (c) 2007-2011 Arista Networks, Inc.  All rights reserved.
# Arista Networks, Inc. Confidential and Proprietary.

import Plugins
import Errdisable

@Plugins.plugin( requires=( 'interface/errdisable/causegroup',
                            'interface/errdisable/cause' ) )
def Plugin( entMan ):
   # Create stp config status and hw.
   entMan.register( 'stp/config', "Stp::Config" )
   entMan.register( 'stp/portMode', "Tac::Dir" )
   entMan.register( 'stp/input/config', "Tac::Dir" )
   entMan.register( 'stp/input/config/cli', "Stp::Input::Config" )
   entMan.register( 'stp/status', "Stp::Status" )
   entMan.register( 'stp/protoStatus', "Stp::ProtoStatus" )
   entMan.register( 'stp/hw', "Stp::Hw" )
   entMan.register( 'stp/counter', "Stp::PortCounterDir" )
   entMan.register( 'stp/standbyStatus', "Stp::StandbyStatus", force=True )
   entMan.register( 'stp/ssoStableControlStatus', "Stp::StableControlStatus" )

   import SysdbPlugin.SysdbLauncher
   agentName = 'Stp'
   agentCfg = { 'name' : agentName,
                'exe' : '/usr/bin/Stp',
                'argv' : [],
                'heartbeatPeriod' : 30 }
   roleName = SysdbPlugin.SysdbLauncher.allSupervisorsRoleName
   SysdbPlugin.SysdbLauncher.agentConfigIs( entMan, roleName, agentCfg )

   entMan.registerLogFacility( 'SPANTREE' )

   # create the bpduguard errdisable CauseStatus entity
   causeDesc = "BPDU received on portfast port."
   Errdisable.ErrdisableCauseGroupInit( entMan, 'bpduguard', False, causeDesc )
```

See e.g., Arista White Paper, EOS: The Next Generation Extensible Operating System (March 2014) (App. M, Ex.

U.S. 7,162,537	Arista Products
	14) at p. 5 ("As figure 3 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change. All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost. Sysdb is very much like an event-driven publish/subscribe model. If the state of an agent changes, Sysdb will send an event notification to that agent, which will then update its local copy. Similarly when the agent writes to the mount point, it only changes its local copy and the write returns immediately. Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent. This centralized database approach to passing state throughout the system, and the automated way the Sysdb code is generated reduces risk and error, improves software feature velocity, and provides flexibility for customers who can use the same APIs to receive notifications from Sysdb to customize and extend switch features."). *See e.g.,* EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code."). *See e.g.,* EOS: An Extensible Operating System (App. M, Ex. 15) at p. 3 ("Sysdb holds all state, while agents perform all processing. Sysdb is an address space that purely holds state and delivers state updates from one agent

U.S. 7,162,537	Arista Products
	to another. For example, when a link goes down, a port driver updates the link state in Sysdb, which delivers the update to the SNMP service, which then sends a trap. Agents may contain copies of Sysdb state for efficiency, but all state is recoverable from Sysdb whenever needed. Like a traditional database engine, Sysdb itself is ultra-reliable because it contains no application code.”).